Dictionary Learning Inspired Deep Network for Scene Recognition

Yang Liu [*1], **Qingchao Chen** [*2], **Wei Chen** [3], **Ian Wassell** [1]
[1]Department of Computer Science and Technology, University of Cambridge, UK
[2]Department of Security and Crime Science, University College London, UK
[3]State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, China
{yl504, ijw24}@cam.ac.uk, qingchao.chen.13@ucl.ac.uk, weich@bjtu.edu.cn

Abstract

Scene recognition remains one of the most challenging problems in image understanding. With the help of fully connected layers (FCL) and rectified linear units (ReLu), deep networks can extract the moderately sparse and discriminative feature representation required for scene recognition. However, few methods consider exploiting a sparsity model for learning the feature representation in order to provide enhanced discriminative capability. In this paper, we replace the conventional FCL and ReLu with a new dictionary learning layer, that is composed of a finite number of recurrent units to simultaneously enhance the sparse representation and discriminative abilities of features via the determination of optimal dictionaries. In addition, with the help of the structure of the dictionary, we propose a new label discriminative regressor to boost the discrimination ability. We also propose new constraints to prevent overfitting by incorporating the advantage of the Mahalanobis and Euclidean distances to balance the recognition accuracy and generalization performance. Our proposed approach is evaluated using various scene datasets and shows superior performance to many state-of-the-art approaches.

Introduction

Scene recognition remains one of the most challenging problems in image understanding due to illumination changes, high intra-class variance, and background occlusion. To tackle these issues, convolutional neural networks (CNNs) (Krizhevsky, Sutskever, and Hinton 2012) (Simonyan and Zisserman 2014) trained on the large scale Places dataset (Places-CNNs) (Zhou et al. 2014) have yielded improved performance for the scene recognition task. The power of CNNs is achieved by learning a strong feature representation in an end-to-end manner for the classification task, instead of hand-crafting features with heuristic parameter tuning. The CNN can be considered as a universal feature extractor, which learns a new representation of the data that permits computationally easier and more effective classifier design. Many deep CNN architectures have been proposed, but fully connected layers (FCL) followed by rectified linear units (ReLu) are now prevalently used to combine the local features extracted from the early convolution and pooling layer. (Sun, Wang, and Tang 2015) observed that the non-linear property inherent in the ReLu unit leads to sparsity in the neural activations, consequently improving the performance of deep learning methods. Normally, for each training image, around half of the neurons in the hidden layer are activated. They argued that moderate sparse on neuron activation can not only make each neuron has maximum discrimination ability, but also make images of different class maximally distinguishable. However, there is no effective mechanism to automatically adjust the sparsity level of the intermediate hidden units based on the training set.

It would be beneficial to take advantage of the sparse model to adjust the sparsity level and enforce different categories to have different subsets of neurons activated, consequently maximizing discriminative power. As an extension of the standard reconstructive dictionary (Aharon, Elad, and Bruckstein 2006), discriminative dictionary learning (DL) methods (Jiang, Lin, and Davis 2013)(Yang et al. 2014) aim to find the optimal dictionary that simultaneously improves the sparse representation and maximizes its discriminative capability. However, the current DL methods cannot achieve state of the art performance in large-scale image classification, especially in scene recognition, in part since most DL models have only been evaluated with traditional handcrafted features, e.g., BOF (Yang et al. 2009) and SIFT (Lowe 2004).

One way to utilize DL to improve scene recognition performance produced by CNN features is to apply DL as a post-processing step trained separately from the training of the CNN, an approach that is adopted in (Xie et al. 2017)(Liu et al. 2014). Arguably, this does not fully harness the strength of DL since it is not integrated with the deep network. It is also worth noting that although the work in (Wang et al. 2016) addresses object detection, it does combine the DL and conventional CNN layers into the end-to-end training framework by simply replacing the softmax classifier by a DL classifier. However, the DL classifier has not been used to replace any fully connected layers before the classifier. In addition, without an explicit parameter for sparsity control of the sparse coding step in the DL classifier, it is difficult to know whether the sparsity of the total network is best optimized and further, to say that the sparseness will be a good regularization for training the low-level layers in the entire

[*]Authors contributed equally.

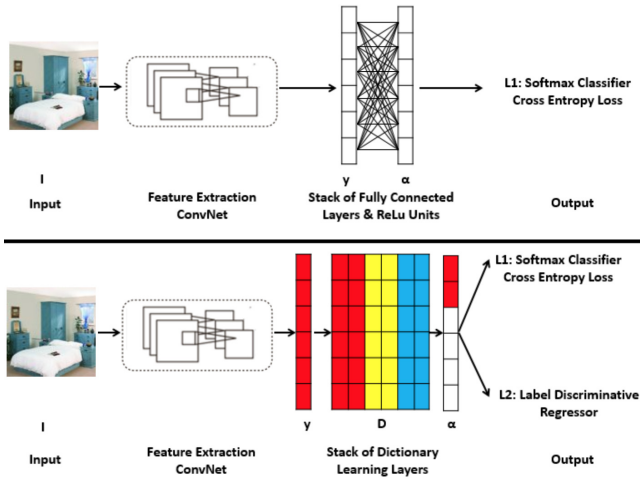

Figure 1: Comparison between the conventional CNN architecture and our CNN-DL architecture. The upper image is the conventional CNN architecture; the lower image is the CNN-DL architecture. We replace the FCL and ReLu units by the DLL, in order to achieve enhanced sparse and discriminative feature representation. The CNN-DL network has two sibling output layers. The first outputs a discrete probability distribution over different categories, computed by a softmax classifier, while the second sibling layer outputs the corresponding discriminant sparse representation that depends upon the dictionary structure.

network. Considering the optimization loss function, it is well known that the cross entropy is more robust to outliers than is Mean-Squared Error (MSE) estimation. However, due to the design of their DL classifier, they have to use the MSE of the reconstruction error, rather than the more robust cross-entropy loss in the conventional classification task. In addition, (Wang et al. 2015) also combined CNN and sparse coding for image super-resolution recovery. Although they cascaded the Learned Iterative Shrinkage-Threshold Algorithm (LISTA) (Gregor and LeCun 2010) network to control the sparsity via an explicit shrinkage function, their loss function is still the MSE loss of the reconstruction error, without any effort to improve feature discriminative capability. To the best of our knowledge, there is no end-to-end learning framework to combine CNN and DL in scene recognition, that can automatically adjust the sparsity level and discriminative capability of the feature representation.

In this paper, our formulation combines the strength of both conventional CNNs and DL procedures in a unified framework, namely CNN-DL. The overall framework for CNN-DL is illustrated in the lower diagram in Fig.1, where arrows indicate the forward propagation direction. Our contributions are summarized as following. Firstly, inspired by the Approximate Message Passing (AMP) (Donoho, Maleki, and Montanari 2009) algorithm and the LISTA network, we design a non-linear dictionary learning layer (DLL) composed of a finite number of recurrent units to integrate the sparse coding and dictionary learning into the deep network architecture. Such a layer can solve the sparse coding with a better convergence rate than LISTA, while also contributing the gradient flow to update the dictionary in each recurrent iteration. In addition, we decoupled the shrinkage function threshold into two factors, so that the threshold is optimally learned to the unknown sparse prior of the neural activation maps. Secondly, we build a new network architecture named as CNN-DL, which replaces the conventional FCLs and ReLu units with a DLL, that exhibits an enhanced sparse and discriminative feature representation. In addition, we take advantage of the structure of the dictionary and design a new label discriminative regressor to further improve the discriminative capability. We argue that such a layer can be used to replace any FC layer and can be cascaded as in the conventional deep learning framework. Thirdly, we propose new constraints to prevent overfitting, i.e., by taking advantage of both the Mahalanobis and Euclidean distances measures, thereby achieving a balance between recognition accuracy for the training set and the generalization performance. Fourthly, we suggest an optimization algorithm for the end-to-end training to jointly learn the parameters in the CNN and DL, that is also compatible with the conventional back propagation scheme. Fifthly, we analyze various factors in CNN-DL that affect the scene recognition performance, including different loss functions and DLL settings. Finally, the CNN-DL model is extensively evaluated using various scene datasets, and it shows superior performance to many state-of-the-art scene recognition algorithms.

Integration of DL and CNN

Network Architecture

Similar to most scene recognition methods based on deep learning networks, our network architecture is designed to learn the transformation from the original image $\boldsymbol{I}_i$ to its corresponding label vector $\boldsymbol{h}_i$ in an end-to-end manner, where i indicates the index of the training samples. The overall network structure is illustrated in the lower image in Fig.1, where arrows indicate the forward propagation direction. The details of each layer are discussed as follows.

To extract the local features of the input image $\boldsymbol{I}_i$, it first goes through the ConvNet, which contains a stack of conventional convolution and pooling layers. To measure the improvements brought by our proposed DLL and the novel loss function in a fair setting, the configurations of all our ConvNet layers are designed based on the same principles and parameter settings used in (Krizhevsky, Sutskever, and Hinton 2012) and (Simonyan and Zisserman 2014). We denote the features obtained from the ConvNet as $\boldsymbol{y}_i = f(\boldsymbol{I}_i) \in \mathbb{R}^M$, where f represents the transformation in the convolution and pooling layers.

Further to obtain the combination of local features, instead of forwarding the features $\boldsymbol{y}_i$ into FCLs (followed by ReLu), we design the non-linear DLLs so as to obtain a sparse feature representation. Each DLL is parameterized with a dictionary $\boldsymbol{D} \in \mathbb{R}^{M \times N}$, composed of a finite number of recurrent units to mimic the sparse coding procedure. Specifically, the dictionary will be ideally learned if the following equation is satisfied for any given extracted feature

representation $\boldsymbol{y}_i$:

$$\boldsymbol{\alpha}_i = arg\min_{\boldsymbol{\alpha}_i} \|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{\alpha}_i\|_F^2 + \lambda \|\boldsymbol{\alpha}_i\|_1 . \quad (1)$$

where $\boldsymbol{\alpha}_i$ is the corresponding sparse feature representation.

In the proposed network, we replace the conventional FCL by the DLL, which is a sub-network aimed at enforcing the sparsity prior on the representation. Conventionally, as in (Krizhevsky, Sutskever, and Hinton 2012), a generic ReLu is used for nonlinear mapping. Since the DLL is designed based on our domain knowledge from sparse coding, we are able to adjust the sparsity level of the neuron activation based on the training set, consequently, obtaining a better interpretation of the layer response. The implementation of this layer should be such that it is capable of passing the error differentials from its outputs to inputs during back-propagation to update the dictionary. The detailed description of the DLL and its relevant optimization rules are discussed in the non-linear dictionary learning layer section.

The design of DLL is not only aimed at a good representing of the given features, its structural flexibility also has the potential to enhance the discriminative capability of the network. This can be achieved by exploiting the dictionary structure and the label information of the input images as part of the final loss function. Inspired by the label consistent K-SVD method (Jiang, Lin, and Davis 2013), we integrate a label discriminative regressor $\mathcal{L}_2 = \sum_i \lambda_1 \|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 + \lambda_2 \left\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{I}\right\|_F^2$ into the final loss objective to measure the discriminative capability of the sparse codes, which forces the training samples from the same class to have similar sparse codes, while forcing samples from different classes to have different sparse codes. The matrix $\boldsymbol{B} \in \mathbb{R}^{N\times N}$ is the optimal linear mapping matrix to transform the original sparse code $\boldsymbol{\alpha}_i$ to the most discriminate sparse feature domain. The $\boldsymbol{q}_i$ represents the ideal discriminative sparse coefficient corresponding to a training sample $\boldsymbol{y}_i$. The nonzero values in $\boldsymbol{q}_i$ only occur at the places where the input training sample $\boldsymbol{y}_i$ and the dictionary atom $\boldsymbol{d}_n$ share the same label. Thus we define the discriminant sparse code $\boldsymbol{q}_i$ as

$$\boldsymbol{q}_i = [q_i^1, q_i^2, ..., q_i^N] = [0...0, 1...1, 0...0]^T \in \mathbb{R}^N. \quad (2)$$

To prevent over-fitting during training, we also add a constraint term on matrix $\boldsymbol{B}$ to balance the Euclidean and Mahalanobis distances as $\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{I}\|_F^2$. The details of the label discriminative regressor and its implementation is introduced in the next section.

Besides utilizing the loss function $\mathcal{L}_2$ to maintain the discriminative sparse code, $\boldsymbol{\alpha}_i$ can also be considered directly as a discriminant feature of original image $\boldsymbol{I}_i$ for classification. Here, a conventional softmax classifier is used to compute the predicted label, and the cross-entropy loss function $\mathcal{L}_1$ is used to compare between the predicted label $\boldsymbol{h}_i^*$ and ground truth label $\boldsymbol{h}_i$. Finally, the overall loss function of our network can be expressed as

$$\min_{\Theta} \mathcal{L}_1 + \lambda_1 \sum_i \|\boldsymbol{q_i} - \boldsymbol{B}\boldsymbol{\alpha_i}\|_F^2 + \lambda_2 \left\|\boldsymbol{B^T}\boldsymbol{B} - \boldsymbol{I}\right\|_F^2 , \quad (3)$$

where Θ represents the parameters within the network, and λ_1 and λ_2 balance the contribution of different terms. We will see in the experiments that the design of the proposed DLL and the novel objective function both contribute to better scene recognition results and smaller model size than a conventional CNN.

Loss Function Design with Integration of the Label Discriminative Regressor

As part of the final loss function in (3), the objective function to improve discrimination can be expressed as

$$\min_{\Theta} \lambda_1 \sum_i \|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 + \lambda_2 \left\|\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{I}\right\|_F^2 . \quad (4)$$

It is worth noting that in the (4), we adopted a new constraint regularized by λ_2 to prevent overfitting. It is designed to incorporate the strength of the Mahalanobis and Euclidean distances, and to achieve balance between the recognition accuracy and the generalization ability. To understand this constraint, we argue that the term $\|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2$ can be regarded as measuring the Mahalanobis distance [1] between the optimal and calculated sparse codes $\boldsymbol{\alpha}_i^*$ and $\boldsymbol{\alpha}_i$ under the transformation matrix $\boldsymbol{B}$, as:

$$\begin{aligned} \|\boldsymbol{q}_i - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 &= \|\boldsymbol{B}\boldsymbol{\alpha}_i^* - \boldsymbol{B}\boldsymbol{\alpha}_i\|_F^2 \\ &= (\boldsymbol{\alpha}_i^* - \boldsymbol{\alpha}_i)^T\boldsymbol{B}^T\boldsymbol{B}(\boldsymbol{\alpha}_i^* - \boldsymbol{\alpha}_i) . \end{aligned} \quad (5)$$

Although the Mahalanobis distance is well known for its discriminative capability, it is prone to be overfitting. Compared with the Mahalanobis distance, the Euclidean distance has a worse ability to discriminate but a better ability to generalize, because it does not take into consideration data correlation across dimensions (Manly 1994). Here, we impose a constraint to ensure that the matrix $\boldsymbol{B}^T\boldsymbol{B}$ has large values on the diagonal and small values elsewhere, so that we can obtain a compromise between the Mahalanobis distance and the Euclidean distance. The constraint is formulated as the Frobenius norm of the difference between $\boldsymbol{B}^T\boldsymbol{B}$ and the identity matrix $\boldsymbol{I}$. More specifically, when λ_2 is small, the Mahalanobis distance should be dominant to measure the distance between $\boldsymbol{\alpha}_i^*$ and $\boldsymbol{\alpha}_i$. In contrast, with the choice of a larger value of λ_2, the matrix $\boldsymbol{B}^T\boldsymbol{B}$ is forced to be close to the identity matrix so that the Euclidean distance is more dominant. In this situation, the Euclidean distance may generalize better to unseen test sets. To sum up, we incorporate the advantage of Mahalanobis and Euclidean distances to maintain the discriminative capability while reducing the overfitting problem.

Nonlinear Dictionary Learning Layer

In this section, we introduce details of the fast yet accurate non-linear DLL which is inspired by the Approximate Message Passing (AMP) algorithm (Donoho, Maleki, and Montanari 2009) and the on-line dictionary learning method in (Mairal et al. 2009). As shown in Fig.2(a), given the input data from previous layers $\boldsymbol{y}$, the DLL is able to compute the final output sparse codes $\boldsymbol{\alpha}^{(K)}$ efficiently using K stacked

[1] The Mahalanobis distance is formulated as $d(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2) = \sqrt{(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)^T\boldsymbol{M}(\boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_2)}$

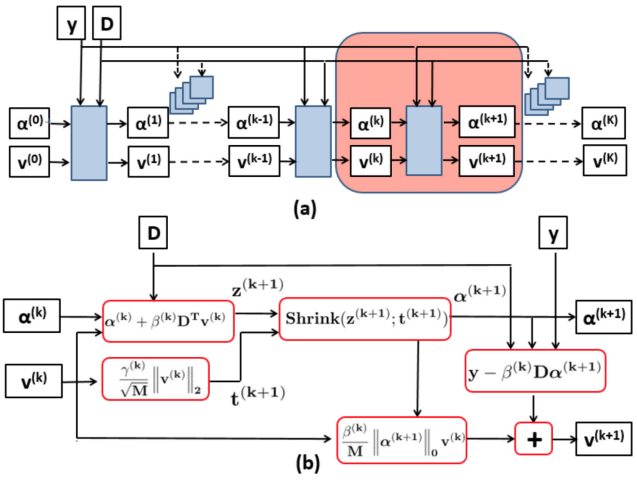

Figure 2: The design of a non-linear dictionary learning layer. Each blue box in the Fig.2(a) represents a recurrent unit, which corresponds to one iteration in the AMP sparse coding procedure. Fig.2(b) represents the operations between adjacent recurrent units, which is shown in the zoomed-in version of the red box in Fig.2(a).

recurrent units (blue boxes), each of which exhibits a similar function to that of an iteration in the AMP algorithm. As shown in Fig.2(b), we represent such adjacent recurrent units in a data flow graph, where the blue box represents the recurrent unit, a node represents a variable and the directed edge represents the flow between two variables. The operations between the adjacent k^{th} and $(k+1)^{th}$ units in the data flow graph can be represented as:

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{\alpha}^{(k)} + \beta^{(k)} \boldsymbol{D}^T \boldsymbol{v}^{(k)} \tag{6}$$

$$\boldsymbol{t}^{(k+1)} = \frac{\gamma^{(k)}}{\sqrt{M}} \left\| \boldsymbol{v}^{(k)} \right\|_2 \tag{7}$$

$$\boldsymbol{\alpha}^{(k+1)} = max(\left| \boldsymbol{z}^{(k+1)} \right| - \boldsymbol{t}^{(k+1)}, 0) \frac{\boldsymbol{z}^{(k+1)}}{\left| \boldsymbol{z}^{(k+1)} \right|} \tag{8}$$

$$\boldsymbol{v}^{(k+1)} = \boldsymbol{y} - \beta^{(k)} \boldsymbol{D} \boldsymbol{\alpha}^{(k+1)} + \frac{\beta^{(k)}}{M} \left\| \boldsymbol{\alpha}^{(k+1)} \right\|_0 \boldsymbol{v}^{(k)} \tag{9}$$

where γ and β represent the learnable scaling factors and $\boldsymbol{t}$ represents the threshold vectors, M represents the dimension of the input feature $\boldsymbol{y}$, the first layer inputs are $\boldsymbol{\alpha}^{(0)} = 0$ and $\boldsymbol{v}^{(0)} = \boldsymbol{y}$. Note that sparse coding steps can be achieved in a relatively small number of iterations to fit existing data sets. We used $K = 2$ recurrent units within each DLL throughout the paper unless otherwise specified. Since the dictionary is shared among recurrent units within one DLL, the parameters in each DLL can then be represented as $\Theta = \{\boldsymbol{D}, \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{k=K}\}$.

There are three major differences between our DLL and the LISTA algorithm: firstly inspired by the AMP, the conventional residual $\boldsymbol{v}^{(k+1)}$ is corrected by use of the Onsager correction (Donoho, Maleki, and Montanari 2009) term: $\frac{\beta}{M} \left\| \boldsymbol{\alpha}^{(k+1)} \right\|_0 \boldsymbol{v}^{(k)}$ in our DLL layer. This correction has been proved to enable the input of the shrinkage function, $\boldsymbol{z}^{(k+1)}$ to be treated as an approximately Additive white Gaussian noise (AWGN) corrupted optimal sparse code. Therefore, many shrinkage functions can be applied easily. Secondly, the threshold selection $\boldsymbol{t}^{(k+1)}$ in our shrinkage function design is decoupled into the scaled factor $\gamma^{(k)}$ and the realization $\boldsymbol{v}^{(k)}$. This enables the threshold of the shrinkage function to be optimized when the prior of sparse codes is not easily modeled. We argue that this will be useful for a deep learning framework, as estimating the prior of the neural activation will be a difficult task. More specifically, learning from training samples, $\boldsymbol{\gamma^{(k)}}$ is adaptive to the unknown and complex prior of the neural activation. Thirdly, except for the dictionary matrix, we do not have to learn an additional transformation matrix (matrix $\boldsymbol{B}$ in the LISTA framework (Gregor and LeCun 2010)), but decouple it into dictionary $\boldsymbol{D}$ and the scaling parameter β. This lowers the model complexity and ensures that the input to the shrinkage function is still essentially AWGN corrupted.

As the overall objective function in (3) does not depend on $\boldsymbol{D}$ explicitly, it is difficult to compute the gradient with respect to $\boldsymbol{D}$ in each recurrent unit. Therefore, we consider the dictionary $\boldsymbol{D}$ as a shared parameter implicitly in each recurrent unit and propose to compute the gradient of the loss function $\mathcal{L}$ with respect to $\boldsymbol{D}$ using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{D}} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(k)}} \frac{\partial \boldsymbol{\alpha}^{(k)}}{\partial \boldsymbol{z}^{(k)}} \frac{\partial \boldsymbol{z}^{(k)}}{\partial \boldsymbol{D}}, \tag{10}$$

where

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(k)}} = \begin{cases} \prod_{k+1}^{K} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(k)}} \frac{\partial \boldsymbol{\alpha}^{(k)}}{\partial \boldsymbol{z}^{(k)}} \frac{\partial \boldsymbol{z}^{(k)}}{\partial \boldsymbol{\alpha}^{(k-1)}} & if\ k < K \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(k)}} & if\ k = K \end{cases} \tag{11}$$

It is also worth noting three points: first, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^{(K)}}$ can be calculated based on the overall objective function (3); Second, the q^{th} element of $\frac{\partial \boldsymbol{\alpha}^{(k)}}{\partial \boldsymbol{z}^{(k)}}$ is set to 0 if the q^{th} element $\boldsymbol{\alpha}_q^{(k)} = 0$, otherwise, it is set to 1. Thirdly, $\frac{\partial \boldsymbol{z}^{(k)}}{\partial \boldsymbol{\alpha}^{(k-1)}}$ can be calculated based on equation Eq (6) and (9) and we use the $\mathcal{L}_1$ norm as an approximation of $\mathcal{L}_0$ norm in Eq (9) to enable standard back propagation in our implementation.

Once $\frac{\partial \mathcal{L}}{\partial \boldsymbol{D}}$ is calculated, the dictionary $\boldsymbol{D}$ in the DLL layer can be updated by stochastic gradient descent. To make this new layer compatible with the other layers in the current CNN framework, we need to consider how the loss function can be back-propagated through this DLL layer to the previous layers. We need to calculate $\frac{\partial \mathcal{L}}{\partial \boldsymbol{y}}$, and once it is obtained, we can perform standard back propagation (Krizhevsky, Sutskever, and Hinton 2012) to update the CNN parameters in the previous layers. In fact, $\boldsymbol{y}$ is similar to the case of dictionary $\boldsymbol{D}$, and can also be obtained by use of the chain rule.

From the perspective of the overall network architecture, different DLLs (cascaded in series) are parametrized by separate dictionaries and all the parameters within the overall network can be trained by standard back-propagation. It will be shown in the experiments that with the integration of the newly proposed DLL, the overall CNN-DL structure is able

to generate better scene recognition performance than can the conventional CNN structure.

Experiments

In this section, we evaluate our approach using various scene recognition datasets and compare with other state-of-the-art approaches. We first introduce the dataset and the parameter settings, and second give an analysis of the factors that effect in the CNN-DL method. Then, we report the recognition performance of the proposed method and compare it with other scene recognition algorithms in the following section. The evaluation is performed for various network architectures and scene datasets.

Datasets and Experimental Settings

Since all the comparison methods only present their results on 15 Scene (Lazebnik, Schmid, and Ponce 2006), MIT Indoor-67 (Quattoni and Torralba 2009), or Sun 397 (Xiao et al. 2010), we employ these three widely used scene recognition datasets in our experiments. We use the average accuracy to evaluate the recognition performance. The parameters in the objective function (3) are determined by 5-fold cross-validation for different datasets as listed in Table. 1. We follow the same training-test partition used in (Liu et al. 2014) (Xie et al. 2017). 15 Scene includes 100 images per class for training and the rest for testing. MIT Indoor 67 includes 80 images of each category for training and 20 images for testing. SUN 397 includes multiple train/test splits, with 50 images per class in the testing set. We present results for the average accuracy over splits.

To completely understand and evaluate effects of different factors in the CNN-DL, a detailed factor analysis regarding parameter selection and layer settings are performed in the next section. For comparisons between other methods, the CNN network architectures we adopt are AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and VGG net (Simonyan and Zisserman 2014). The model is first trained on the large auxiliary dataset with image level supervision, including ImageNet data (Krizhevsky, Sutskever, and Hinton 2012) or Place205 data (Zhou et al. 2014). To adapt the pre-trained CNN to the new scene dataset, we perform domain-specific fine-tuning on the three scene datasets respectively. However, in practice, it is often not possible to fine tune all the parameters in the network because of the limited data, leading to over-fitting. So in the following experiments, we only replace last two FCLs by an equivalent number of the proposed non-linear DLLs. It worth noting that the number of dictionary atoms per class should be one in the final DLL. Detailed information about the dictionary size of the previous to last DLL layer are shown in Table.1. To balance the trade-off between speed and accuracy, we used $K = 2$ recurrent units within each DLL throughout the paper unless otherwise specified.

Factor Analysis

In this section, we investigate how the performance of CNN-DL is affected by different factors using the MIT Indoor 67 dataset. All the analysis in this section is based on results

Table 1: Dataset and Experimental Details

Dataset	Dictionary Size	λ_1	λ_2
15 Scene	600 atoms (40 atoms/class)	0.3	0.001
MIT Indoor 67	2680 atoms (40 atoms/class)	0.5	0.005
SUN 397	3970 atoms (10 atoms/class)	0.3	0.02

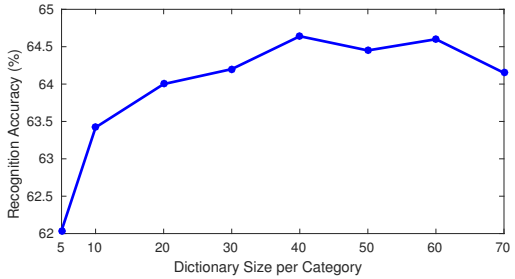

Figure 3: Effect of dictionary size on the recognition accuracy.

achieved by CNN-DL on AlexNet (pre-trained on ImageNet data). We will discuss two main factors, including the configuration of the DLL and the use of different loss functions.

Factor 1: Proposed Dictionary Learning Layer

In this section, we verify the effectiveness of DLL components in different configuration settings, including dictionary size (factor 1.1), the number of recurrent units (factor 1.2) and the use of cascaded DLLs. In some of the tests, we also compare the performance of DLL with its conventional alternative FCL+ ReLu. Note that in order to remove the improvement brought by the novel objective function design, we use the simple cross entropy loss $\mathcal{L}_1$ directly to enable a fair comparison.

Factor 1.1: Different Dictionary Size in DLL

It is worth noting that an important factor within the DLL is the dictionary size, as it controls the model capacity. As shown in (Jiang, Lin, and Davis 2013)(Yang et al. 2014), the larger the size of the dictionary, the better is the performance of the traditional DL model. However, as input of the DLL is the feature learned by the deep network, it is still unclear what dictionary size fits best to these high level features. In general, the ideal DL method should achieve an acceptable level of performance using a relatively small dictionary size.

In this section, we use the MIT Indoor 67 dataset as an example for the evaluation and analysis. Specifically, we only replace the conventional FCL7 by our DLL followed by FCL8 as the classifier. For each class, we randomly choose 80 images for training and the rest for testing, similar to the experimental setting in (Liu et al. 2014). The parameters of the dictionary are initialized using a truncated normal distribution, with the number of dictionary atoms per class varying from 5 to 70. Fig.3 shows that the performance of the CNN-DL method improves when the dictionary size increases and reaches a maximum at around 40 atoms per category. This may relate to the diversity of training data under the particular category, while too many dictionary atoms per class may introduce information redundancy or noise into the feature representation.

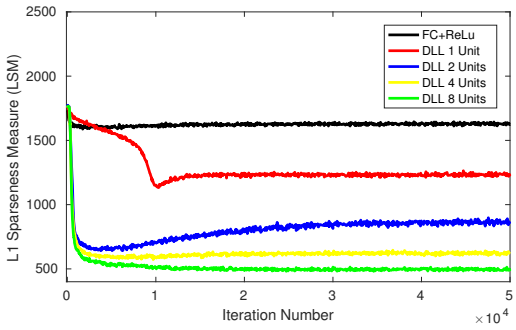

Figure 4: LSM of DLL activations using different layers

Table 2: Effect of number of Recurrent Units in DLLs

Loss function	Layer Setting	No.Units	Recognition Rate
$\mathcal{L}_1$	FCL7+FCL8	-	61.80
$\mathcal{L}_1$	FCL7+DLL8	1	63.21
$\mathcal{L}_1$	FCL7+DLL8	2	64.63
$\mathcal{L}_1$	FCL7+DLL8	4	64.32
$\mathcal{L}_1$	FCL7+DLL8	8	63.08

Factor 1.2: Number of recurrent unit in DLL

In this section, we evaluate the effect of the number of recurrent units in the DLL. Specifically, we only replace the last FCL with a DLL in the experiments. As shown in Figure 4, we plot the $\mathcal{L}_1$ sparseness measure (LSM) (Li et al. 2017) of the conventional FCL output and the DLL output based on using 1,2,4 and 8 recurrent units. These LSM losses are based on inputting the same test datasets and calculating the $\mathcal{L}_1$ norm of their final outputs. It can be clearly shown that output of the FCL is less sparse than others and with the increasing number of recurrent units, the DLL outputs are more sparse. Further, for the different sparseness levels, we can evaluate the corresponding recognition rates as shown in Table. 2. The recognition rate increases and achieves the best rate of 64.63% when the number of recurrent units is two. As more units are utilized, the recognition rate falls to 63.08%, but is still better than using the FCL only. This experiment shows that only appropriate levels of sparseness can lead to good recognition rates.

Factor 1.3: Usage of Cascaded DLLs

Here, we analyze the effect of cascaded DLLs in the deep CNN framework by using different layerwise configurations, as shown in Table.3. For a fair comparison, we used 2 recurrent units, fixed the dictionary size to 2680. It is worth noting that, as the LISTA network also involves sparse coding, we compare our DLL with LISTA in this section. As shown in Table.3, replacing FCL7 and FCL8 by two LISTA layers improves performance from 61.79% to 63.90% and replacing a single FCL with our proposed DLL increases performance to around 64.46% and 64.63%, that is already 0.5% improvement to the LISTA layers. Finally, the recognition rate rises to 65.2% when we replace both FCL layers with DLLs, which is around a 3.5% improvement compared to FCLs only. This experiment shows the superiority of our approach compared with LISTA and FCL layers, and also shows that using two DLLs outperforms the use of one

Table 3: Analysis for Factor1.3: cascaded DLLs

Loss function	Layer Setting	Recognition Rate
$\mathcal{L}_1$	FC7+FC8	61.80
$\mathcal{L}_1$	DLL7+FC8	64.46
$\mathcal{L}_1$	FC7+DLL8	64.63
$\mathcal{L}_1$	DLL7+DLL8	65.19
$\mathcal{L}_1$	LISTA7+LISTA8	63.90

Table 4: Recognition Rates (%) for different factors setting

Loss function	Layer setting	Recognition Rate
$\mathcal{L}_1$	FC7+FC8	61.80
$\mathcal{L}_1$	DLL7 +DLL8	65.19
$\mathcal{L}_1 + \mathcal{L}_2$ (-)	DLL7 +DLL8	66.11
$\mathcal{L}_1 + \mathcal{L}_2$	DLL7 +DLL8	66.60

DLL. The reason that our DLL outperforms LISTA may be owing to the decoupling of the learned threshold into two factors, that is able to adapt the threshold in the shrinkage function of the DLL under the unknown sparse prior of the activation outputs. In addition, due to the fact that LISTA needs to learn an extra matrix in the sparse coding procedure while our DLL does not, our model can be trained more efficiently with fewer parameters. To sum up, the new proposed DLL contributes in yielding better scene recognition performance.

Factor 2: Effect of different terms in the Objective Function

We will now investigate how the performance of CNN-DL is affected by the design of the objective function. We evaluate the loss function with and without the label discriminative regressor of the sparse code and the corresponding constraint. A comparison of the recognition performances are shown in the Table.4[2]. We can see that with the help of the label discriminative regressor, the performance is consistently better than using only the simple softmax function. It verifies that the adoption of the novel discrimination term takes advantage of the structure of the dictionary in the DLL, consequently enhancing the discriminative capability of the overall network. In addition, the loss function with the constraints always shows superior performance to those without it. This shows that the constraint we developed for preventing over-fitting in training is effective.

Results and Comparisons

In this section, we compare our CNN-DL method with other existing scene classification approaches. The classification performance metric is the percentage of correctly classified test data. The benchmark algorithms for comparison are Hybrid-CNN (Zhou et al. 2014), Multiscale orderless pooling CNN (MOP-CNN)(Gong et al. 2014), Semantic Fisher vector CNN (SFV) (Dixit et al. 2015), hybrid CNN and dictionary-based model (CNN-DBL) (Xie et al. 2017), Unified representative and discriminative learn-

[2] $\mathcal{L}_1 + \mathcal{L}_2$ (-) represents using $\mathcal{L}_1 + \mathcal{L}_2$ as the objective function without the term to prevent overfitting

Table 5: Recognition Rates (%) for different architectures and multi-scale variations

Architecture	Pretrain Set	No. scales	15 Scenes		MIT Indoor 67		SUN 397	
			Alex	VGG	Alex	VGG	Alex	VGG
Baseline	IN	1	86.50	89.89	61.80	71.11	46.33	54.09
CNN-DL	IN	1	88.33	92.90	66.60	78.33	51.67	60.23
Baseline	PL	1	89.96	91.20	72.88	79.45	57.07	65.10
CNN-DL	PL	1	91.30	93.30	76.10	82.86	60.82	67.90
Hybrid-CNN	IN+PL	1	53.86	–	70.80	–	53.86	–
DAG-CNN	IN	1	–	91.90	–	77.50	–	56.20
DSP-CNN	IN	1	–	91.78	–	78.28	–	59.78
Dual	IN	2	92.16	93.84	71.87	79.04	56.62	61.07
Dual CNN-DL	IN	2	92.31	94.54	**76.56**	83.37	**59.03**	65.20
Dual	PL	2	93.80	95.18	76.87	83.43	62.60	67.59
Dual CNN-DL	PL	2	94.20	96.03	80.30	86.43	66.10	70.13
MOP-CNN	IN	3	–	–	68.88	–	51.98	–
CNN-DBL	IN	3	–	–	74.09	82.24	57.31	64.53
MFAFV-NET	IN/PL	3	–	–	75.01	82.66	57.15	64.59
URDL	IN	4	91.15	–	71.90	–	–	–
SFV	IN	4	–	–	72.86	–	54.40	–
SFV+PLACES	IN/PL	4+1	–	–	79.00	–	61.72	–

ing model (URDL) (Liu et al. 2014), Directed acyclic graph CNN (DAG-CNNs)(Yang and Ramanan 2015), Deep spatial pyramid CNN (DSP-CNN)(Gao et al. 2015) and Mixture of factor analyzers Fisher vector network (MFAFV-NET)(Li, Dixit, and Vasconcelos 2017). Detailed information about each model is shown in Table.5, including the pre-train auxiliary dataset and the number of scales.

More specifically, we first train the CNN based network on the auxiliary data, including ImageNet data (IN) or Place205 data (PL), then fine tune the network as described previously for different scene datasets. By replacing the FCLs with the proposed DLLs, we can evaluate the performance achieved by the CNN-DL method. It can be seen that the network pre-trained by PL always performs better than the counterpart pre-trained by IN, which corresponds with the findings in (Zhou et al. 2014). In addition, the VGG network architecture always demonstrates superior performance to the Alex network architecture owing to the exploitation of deeper models. When evaluating various methods on a single global scale, our CNN-DL method consistently outperforms the other CNN based scene recognition methods, which justifies the effectiveness of integrating the DLL into the network architecture. This may be because the DLL considers sparsity control of the neuron activation and the discrimination capability simultaneously. Furthermore, usage of the new constraint in (4) is able to prevent overfitting, balancing between the recognition accuracy of the training set and the generalization performance.

It worth noting that among the comparison methods, the CNN-DBL and URDL methods are based on dictionary learning models, however, these approaches apply the dictionary learning model as a post-processing step disconnected from the training of CNN, and so do not fully harness the strength of DL since it is not integrated with the deep network. As these methods show improved performance when evaluated on multiple scales (Xie et al. 2017) (Liu et al. 2014), we also consider the multiple scale case for our CNN-DL method. In this paper, we evaluate pairwise combinations of CNNs used at two different scales, represented as Dual CNN-DL. As in (Herranz, Jiang, and Li 2016), the dual architecture consists of two CNNs processing images at two scales. We regard this as the baseline for the two scales evaluation. Instead of concatenating the two resulting final FCL activations into a feature and training the SVM as in (Herranz, Jiang, and Li 2016), we concatenate the sparse feature representation into a feature and train the SVM in this new discriminant latent space. The results in Table.5 show a considerable improvement in the performance of Dual CNN-DL (Pre-trained on IN), achieving an accuracy of 76.56% on MIT Indoor 67 and 59.03% on SUN 397 for only two scales. Compared with the other dictionary learning based models, our method obtains significantly better performance on MIT Indoor 67 and SUN 397, while using only two scales compared with 3 or 4 scales in CNN-DBL and URDL. In this work, we only combine two scales in a dual architecture for CNN-DL, though it is also possible to consider the combination of more scales at the cost of a more complex architecture.

Conclusion

In this paper, we replace the conventional FCL and ReLu by our proposed nonlinear DLL. The proposed approach is capable of controlling the sparsity of the neuron activation in the forward pass, while passing the error differentials from its outputs to inputs during back-propagation to update the dictionary. Our CNN-DL architecture takes advantage of the potential structure of the dictionary, in order to harness the discriminant capability of the features via a new label discriminative regressor. In addition, we propose new constraints to prevent overfitting, by incorporating the advantage of the Mahalanobis and Euclidean distances and balancing the recognition accuracy and generalization performance. We suggest an algorithm to train the whole network end-to-end by performing conventional back propaga-

tion and avoiding offline post-processing. The superior performance of the proposed method in comparison to the state-of-the-art is demonstrated using various scene datasets.

References

Aharon, M.; Elad, M.; and Bruckstein, A. 2006. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing* 54(11):4311–4322.

Dixit, M.; Chen, S.; Gao, D.; Rasiwasia, N.; and Vasconcelos, N. 2015. Scene classification with semantic fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2974–2983.

Donoho, D. L.; Maleki, A.; and Montanari, A. 2009. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences* 106(45):18914–18919.

Gao, B.-B.; Wei, X.-S.; Wu, J.; and Lin, W. 2015. Deep spatial pyramid: The devil is once again in the details. *arXiv preprint arXiv:1504.05277*.

Gong, Y.; Wang, L.; Guo, R.; and Lazebnik, S. 2014. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, 392–407. Springer.

Gregor, K., and LeCun, Y. 2010. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 399–406.

Herranz, L.; Jiang, S.; and Li, X. 2016. Scene recognition with cnns: objects, scales and dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 571–579.

Jiang, Z.; Lin, Z.; and Davis, L. S. 2013. Label consistent K-SVD: Learning a discriminative dictionary for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(11):2651–2664.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Lazebnik, S.; Schmid, C.; and Ponce, J. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, 2169–2178. IEEE.

Li, J.; Zhang, T.; Luo, W.; Yang, J.; Yuan, X.-T.; and Zhang, J. 2017. Sparseness analysis in the pretraining of deep neural networks. *IEEE transactions on neural networks and learning systems* 28(6):1425–1438.

Li, Y.; Dixit, M.; and Vasconcelos, N. 2017. Deep scene image classification with the mfafvnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5746–5754.

Liu, B.; Liu, J.; Wang, J.; and Lu, H. 2014. Learning a representative and discriminative part model with deep convolutional features for scene recognition. In *Asian Conference on Computer Vision*, 643–658. Springer.

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60(2):91–110.

Mairal, J.; Bach, F.; Ponce, J.; and Sapiro, G. 2009. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, 689–696. ACM.

Manly, B. F. 1994. *Multivariate statistical methods: a primer*. CRC Press.

Quattoni, A., and Torralba, A. 2009. Recognizing indoor scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 413–420. IEEE.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sun, Y.; Wang, X.; and Tang, X. 2015. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2892–2900.

Wang, Z.; Liu, D.; Yang, J.; Han, W.; and Huang, T. 2015. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision*, 370–378.

Wang, K.; Lin, L.; Zuo, W.; Gu, S.; and Zhang, L. 2016. Dictionary pair classifier driven convolutional neural networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2138–2146.

Xiao, J.; Hays, J.; Ehinger, K. A.; Oliva, A.; and Torralba, A. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, 3485–3492. IEEE.

Xie, G.; Zhang, X.; Yan, S.; and Liu, C. 2017. Hybrid cnn and dictionary-based models for scene recognition and domain adaptation. *IEEE Transactions on Circuits and Systems for Video Technology* 27(6):1263–1274.

Yang, S., and Ramanan, D. 2015. Multi-scale recognition with dag-cnns. In *Proceedings of the IEEE International Conference on Computer Vision*, 1215–1223.

Yang, J.; Yu, K.; Gong, Y.; and Huang, T. 2009. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 1794–1801. IEEE.

Yang, M.; Zhang, L.; Feng, X.; and Zhang, D. 2014. Sparse representation based Fisher discrimination dictionary learning for image classification. *International Journal of Computer Vision* 109(3):209–232.

Zhou, B.; Lapedriza, A.; Xiao, J.; Torralba, A.; and Oliva, A. 2014. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, 487–495.